[0074] The class declaration for the Monitor class may thus be expressed as:

MONITOR CLASS - TABULATED VIEW
CLASS NAME: Monitor

ATTRIBUTES	Type	Comments
screen_resolution	integer	

refresh_rate	Value	integer
		warp_master_value: ..\..\pixel_clock
		warp_rule:
		If warp_master_value > '100', max = '50';
		If warp_master_value > '200', max = '60';
		If warp_master_value > '300', max = '70';

[0075]

MONITOR CLASS - PSEUDO-CODE

```
class Monitor;
{
        attributes:
        {
                refresh_rate:
                {
                        type: integer;
                        warp_master_value: ..\..\pixel_clock;
                        # define warp_master_value
                        warp_rule: # define warp_rule
                        {
                                If warp_master_value > '100', max = '50';
                                If warp_master_value > '200', max = '60';
                                If warp_master_value > '300', max = '70';
                        }
                }
                screen_resolution: # define the characteristics of the
                                        variable screen_resolution
                {
                        type : integer;
                }
        }
}
```

[0076] An overview of this value warp functionality is shown in **FIG. 5***a*. For example, when the accessor method used to retrieve the value of the refresh rate attribute is called (step **500**), it is determined whether the attribute has any associated warp information (step **502**). If there is no warp information, the value of the attribute is returned (step **510**). If there is warp information available the warp master value is retrieved (**504**) as previously described using, for example, the tree navigation functionality. Any warp rule information is then obtained (step **506**) and is applied to the warp master value (step **508**). The attribute value is then returned (step **510**).

[0077] To ensure that subsequent changes to the warp master value will cause a change in the warp value a registration mechanism may be used to register the reference of the warp value, along with the warp rule, with the warp master value. This may be achieved, for example, by declaring the warp master value as of a class type having appropriate data repositories and methods for interpreting and performing the required functionality. Thus, when a warp master value is interrogated, the reference to the warp value, along with any associated warp rules, are registered with the warp master value. This step may be added, for example, between the steps **504** and **506** of the flow diagram of **FIG. 5***a*.

[0078] Thus, should the warp master value be subsequently modified, the accessor methods used for modifying the warp master value will check to see whether any warp values have been registered therewith, and if so will update the registered warp value directly using the registered warp rules, as outlined in the flow diagram of **FIG. 5b**. When the accessor method of an attribute is accessed, a check is made to determine whether there is any warp information registered with the attribute (step **520**). If no such information is registered, the attribute may be modified, for example, in the usual manner (step **528**). If warp information is registered, the reference of the attribute which is registered thereat is obtained (step **522**), along with the warp rule (step **524**) which may also have been registered. In the event that more than one warp rule has been registered the appropriate warp rule can be selected. The warp rule is then applied and the attribute which is registered is modified directly using the registered reference (step **526**). Finally, the attribute within the class may be modified (step **528**).

[0079] As previously mentioned, use of Class::MethodMaker in Perl causes objects which use Class::MethodMaker to be dynamically created as they are accessed. **FIG. 6** is a flow diagram outlining one way in which the main steps may be performed when using a Perl-type implementation for requesting a warp value from an object which has yet to be created.

[0080] The request for the value of the refresh rate attribute **112** is made through the monitor object **108**, for example, by calling the appropriate accessor function. If the monitor object does not exist (step **604**), then it is created (step **606**). At step **608** it is determined whether the requested attribute is a warp value. If the requested attribute is a warp value, then the reference of the warp master object is retrieved (step **610**) as previously described. The warp value and warp rules are registered, or stored, within the warp master object (step **612**) as described above, and a check is made to see whether the warp master value exists and has been previously defined (step **614**). If the warp master value has been defined, the warp rule is applied to the warp master value (steps **616** and **618**) and the warp value is returned (step **624**).

[0081] A second type of dependency is where a class is dependent on an attribute of another class (the warp master). For example assume that the computer type attribute **103** of the computer object **102** indicates whether the computer is a laptop or a desktop computer. A desktop computer may thus have a computer monitor, not an LCD screen, whereas a laptop computer may have an LCD screen, not a computer monitor. Thus, depending on the value of the computer type attribute **103**, either a monitor object or an LCD screen object should be created to enable the correct configuration behavior to be modeled.

[0082] In an embodiment of the present invention, this functionality is implemented through use of a 'hidden' intermediate object hereinafter referred to as a warp object, as shown in **FIG. 4**. In the present example, the warp object **105** lies intermediate the video card object **104** and a monitor or LCD screen object, **108** and **116** respectively, and effectively regulates access to the underlying objects, as will